Web 7.0 Application Note (#web7, @web7arch)

Web 7.0 "DIDFax" Windows Printer Driver Scenario

Version 0.1 – December 9, 2022

Michael Herman, Self-Sovereign Blockchain, Futurist, Architect, and Developer

Trusted Digital Web, Hyperonomy Digital Identity Lab, Parallelspace Corporation

Alberta, Canada

mwherman@parallelspace.net

CONTEXT

Purpose

The purpose of this application note is to describe the "DIDFax" Windows Printer Driver within the context of the cardiac specialty physician's clinical setting. The "DIDFax" Windows Printer Driver scenario uses DIDComm Messages, DIDComm Message Attachments, and DIDComm Agents using Layer 6 of the Web 7.0 DIDComm-ARM model[1].

Goals

This document was produced to address the following goals:

- Illustrate how a DIDComm Agent-based software system can deliver high business value by solving a communications problem within the healthcare ecosystem.
- Illustrate how Web 7.0 DIDComm-ARM models and DIDComm Notation can be used to assist in the design and visualization of DIDComm Agent-based software systems.

Intended Audience

The intended audience for this whitepaper is a broad range of professionals interested in furthering their understanding of decentralized identity concepts for use in software apps, agents, and services. This includes software architects, application developers, and user experience (UX) specialists; as well as people involved in a broad range of standards efforts related to decentralized identity, Verifiable Credentials, and secure storage.

SCENARIO

The inspiration was a visit to my cardiac specialist yesterday. The cardiologist has an internal system he uses for recording his doctor notes but they never leave his system (and hence, his clinic) unless he prints

[1] Web 7.0 DIDComm Agent Architecture Reference Model (DIDComm-ARM) (https://hyperonomy.com/2022/12/07/web-7-0-didcomm-agent-architecture-reference-model-didcomm-arm/)

them out and faxes them. Consequently, my general physician rarely sees the cardiologist's notes unless the cardiologist makes a special effort to fax the notes to my general physician. Recently, we (collectively) had an important breakdown in communication (and treatment) as a result.

SOLUTION CONCEPT

The following solution concept provides a more general solution to a broader problem of being able to securely and privately share information locked up in Windows desktop applications (e.g. Microsoft Office (Word, Excel, PowerPoint, Outlook, Visio, etc.), any Web Browser (Microsoft Edge, Chrome, Firefox, etc.), business applications, graphics art and design applications, computer-aided design and modeling, enterprise architecture modeling tools, banking and financial applications, etc.

The solution concept makes use of the common metaphor of simply printing a document using the File→Print functionality universally available in every Windows desktop application. The solution concept is illustrated in the following figure.

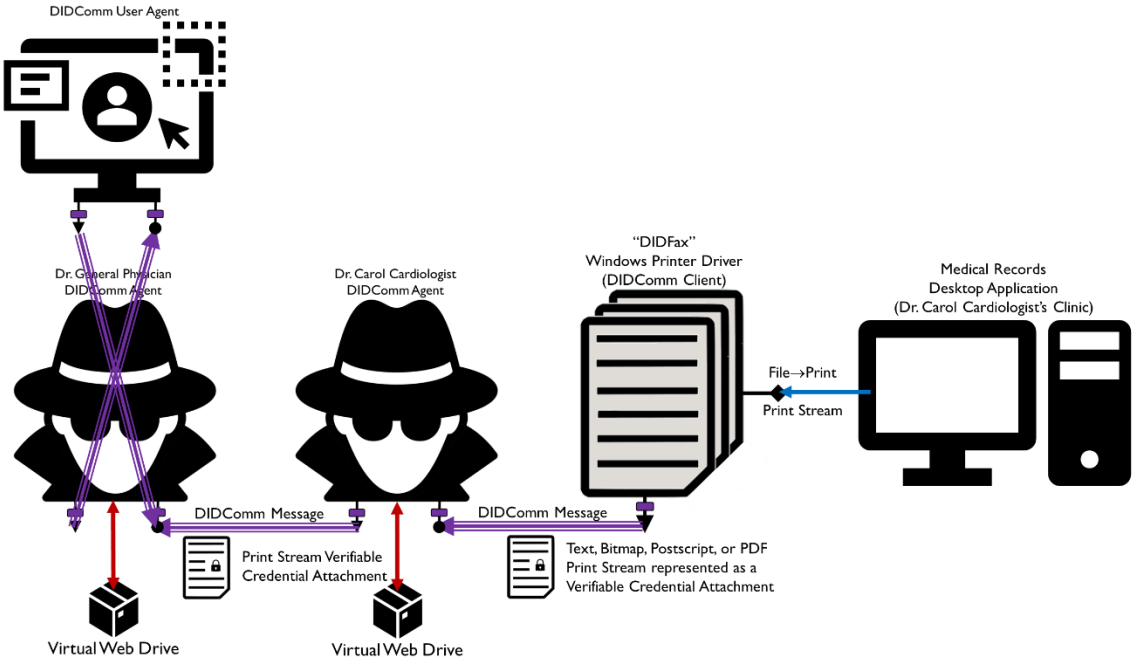

Figure 1. "DIDFax" Windows Printer Driver Solution Concept

The flow is from right to left in the above model:

1. Dr. Carol (the cardiologist) enters her notes into her clinic's medical records desktop application. Note: The solution concept works equally well for cloud and web browser-based applications.
2. When Carol has completed entering and saving her notes, Carol clicks File→Print to "DIDFax" a copy of the notes to the patient's general physician.
3. Carol completes the DIDFax of her notes by selecting the "DIDFax" printer driver in the drop-down list of printers that have been configured into her Windows laptop or desktop computer. Carol clicks OK to confirm her printer selection and to initiate the DIDFax process.

4. The DIDFax printer driver queries Carol's DIDComm Agent (and indirectly, her Virtual Web Drive) to, in effect, retrieve and display a list of Carol's physician contacts indexed by each physician's decentralized identifier (DID).
5. Carol selects the patient's physician from the drop-down list and clicks OK.
6. The DIDFax printer driver formats the information to be "printed" and forwarded to the patient's general physician using the format Carol has selected (e.g. text, bitmap, PostScript, PDF, etc.).
7. The formatted information is serialized into a Verifiable Credential format.
8. Carol's DIDComm Agent attaches the Verifiable Credential to a new DIDComm Message that is addressed to the DID of the patient's general physician.
9. In the Outbound Processing step of this process, the DIDComm Message (and attached Verifiable Credential) is signed using the Sender's Private Key (Dr. Carol Cardiologist's Private Key) and then encrypted using the Public Key of the Receiver (Dr. General Physician, the patient's physician).
10. The signed and encrypted DIDComm Message (and attached Verifiable Credential) is then transported to and received and accepted by the patient's general physician's DIDComm Agent.
11. The general physician's DIDComm Agent decrypts and authenticates the inbound DIDComm Message and extracts the Verifiable Credential containing the serialized print stream representing Carol's doctor notes.
12. The general physician's DIDComm Agent then most likely stores the incoming DIDComm Message (and attachment) in the agent's local Virtual Web Drive.
13. When Dr. General Physician is available, he/she can use the DIDComm User Agent associated with "medical records" to request, display, interact, and possibly, reply to Dr. Carol Cardiologist's doctor notes regarding their mutual patient. NOTE: The incoming DIDComm Message (and attachment) from Carol now lives in the Virtual Web Drive attached to Dr. General Physician's DIDComm Agent.

NOTE: The above solution concept is made possible by using DIDComm, DIDComm Messaging, and DIDComm Agents (as described in the DIDComm-ARM). The above solution concept is not achievable using the classical notion of a digital identity wallet.

REFERENCES

- Web 7.0 DIDComm Agent Architecture Reference Model (DIDComm-ARM) (https://hyperonomy.com/2022/12/07/web-7-0-didcomm-agent-architecture-reference-model-didcomm-arm/)

COPYRIGHT